MAPPING MARMOT PROCESS MODEL INTO PECOS COMPONENT INFRASTRUCTURE FOR EMBEDDED REAL-TIME SYSTEM

Suzila Sabil, Dayang Norhayati Abang Jawawi and Shahliza Abd Halim
Department of Software Engineering, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia.
suzy_sbl@yahoo.com, dayang@utm.my, shahliza@utm.my

ABSTRACT

Embedded software development requires multidisciplinary knowledge including software, mechanical and electronic engineering fields. Engineers struggle hard to master the pitfalls of modern, complex embedded systems, but often they only approach the problems from their individual perspectives. What is really lacking is a vehicle to transport the recent advances in software engineering and component technologies into the embedded world in a way that engineers of the three disciplines can actually communicate and understand each other. Beside that, in software development of embedded real time (ERT), software functionality is not the only focus but non-functionality such as timing, resource constraint etc is also an important focus. This will increase the complexity of the software development for ERT system. To meet these challenges, ERT software developer must be able to cope with complexity, communicate and understand each knowledge involves, to quickly adapt to the changes and capable to support extra-functionality. A solution approach receiving increased attention is components-based software for ERT systems. Therefore, methodological support is important to enable systematic development of component based software. This paper will propose integrate MARMOT model and PECOS component model as a methodological support, whereby MARMOT is used to solve multidisciplinary knowledge and PECOS to solve multi constrain (focus on timing).

Keywords—Component-based software and embedded real-time systems

1. INTRODUCTION

In software development of embedded real time (ERT) system, software functionality is not the only focus but extra-functionality such as timing, resource constraint etc is also an important focus. This will increase the complexity in software development for ERT system (Yen, et. al, 2002) (Martin, et. al, 2007). To meet these challenges, ERT software development must be able to cope with complexity, to adapt quickly to changes and capable to support extra-functionality. Existing industrial component model technologies such as OMG's CORBA Component Model (CCM), Microsoft's (D)COM/COM++, .NET, SUN Microsystems' JavaBeans and enterprise JavaBeans, are not suitable to develop ERT systems because they do not address the non-functional properties in ERT systems. To meet the requirement of ERT systems, a number of component technologies such as PBO (Stewart, et. al, 1997), Koala (Ommering, et. al, 2000), PECOS (Nierstrasz, et. al, 2002), KobrA (Colin Atkinson, et. al, 2000) with extension of MARMOT method to support ERT system and ReFlex (Wall, et. al, 2003) have emerged. All of these ERT component models have their own unique strengths to support their nature of ERT problem domain. PBO was developed for control of manipulator robotics, Koala for consumer products, KobrA is for software product line, PECOS for field devices systems and ReFlex for product-line real-time systems.

Beside the focus on non-functional properties embedded software development requires multidisciplinary knowledge including software engineering, mechanical and electronic fields. Engineers struggle hard to master the pitfalls of modern, complex embedded systems, and often they only approach the problems from their individual perspectives (Bunse, et. al, 2007). What is really lacking is a vehicle to transport the recent advances in software engineering and component technologies into the embedded world in a way that engineers of the three disciplines can actually communicate and understand each other (Bunse, et. al, 2006). Therefore, Method for Component-Based Real-Time Object-Oriented Development and Testing (MARMOT) has emerged as a development method for mastering multi-disciplinary (involving mechanical, electronic and software engineering) embedded systems

development. MARMOT has more features that are particularly important in embedded real time development system such as hardware integration, real time specification, aspect orientation and scheduling (Atkinson, et. al, 2002). MARMOT is based upon fundamental principles, such as software or hardware integration, real time specification, aspect orientation and scheduling that are fully in line with the KobrA method. A weakness of the KobrA method is it does not have capability to support the non-functional requirements (Khan, et. al, 2006). Through MARMOT has the advantages of focusing on multi-disciplinary knowledge and can support non-functional requirement . From the previous evaluation MARMOT still has two disadvantages. The first one is it cannot support software component at details level. The second one it has no method interfaces for component communication.

Accordingly to overcome weaknesses of MARMOT method, PECOS component technology will be used to support software component at details level and method interfaces. Jawawi, at, al, (2004) identified the strengths of PECOS component model for ERT systems among them are light-weight component model, industrial recognized component model, support of development of platform independent component based systems and support predictable real-time performance ERT system.

Component technologies especially from the major organisation such as Microsoft, Sun and OMG are becoming widespread and better known. Nowadays, component technologies allow us to specify a component using a common language, component middleware and component architecture. However, methodology hasn't caught up on the move towards component software (Poels and Dedene, 2000). Technology alone is not enough to build the component software system, thus methodological support also equal importance. Therefore, this paper propose process model beginning from analysis phase, early design phase, detail design phase, integration, composition right to implementation phase.

Overview of MARMOT method will describe in Section 2. Section 3 will elaborate briefly of PECOS component model. Meanwhile, Section 4 describes the process model that proposed and the implementation of case study will describe in Section 5. Meanwhile, Section 6 will be discussing and concluding the process model to support component base software development for ERT system.

2. OVERVIEW OF MARMOT METHOD

MARMOT stands for Method for Component-Based Real-Time Object-Oriented Development and Testing (MARMOT). The growing complexity of the system requires well-structured approaches, which in turn increases the possibilities for reusability. Reuse can be seen as a major driving force in hardware and software development. The MARMOT method is introduced to facilitate component-based structuring and reused in embedded systems development. MARMOT is an extension to the KobrA method. MARMOT adds concepts addressing the specific requirements of developing embedded systems. Composition is a key activity in component-based development with MARMOT. A system is viewed as a hierarchy of components, in which the parents/child relationship represents composition. The component within an embedded system belongs to one out of three groups or types: 1.Software 2.Hardware (divided into electronics, mechanics and mechatronics components) 3.Software/hardware components (Bunse, et. al, 2006). Software and hardware components are treated in the same logical way. In principle, hardware components in an embedded system typically consist of the hardware itself and a device driver to communicate with the software. For CBD, additional interface definition is required which follows the same standard as that for software components. Therefore, a "hardware wrapper" must be devised to provide such an interface, and the one that triggers the events concerning hardware interruptions, and passes calls and parameter's device driver. The wrapper and the device driver hide hardware-specific details and allow the component to participate in remote method calls (Bunse, et. al, 2006).

2.1 MARMOT Process Model

The core principle of MARMOT is separation of concerns, as it associates its main development effort with two basic dimensions that map four basic activities. The two dimensions can be show in Figure 1. The two dimensions are composition/decomposition dimension and abstraction/concretisation dimension.

MARMOT process model include 4 activities, and they are decomposition, embodiment, composition and validation. Descriptions of the four activities are discussed by (Bunse, 2006) to complete the overall process development system for embedded real time system as provided in MARMOT method.

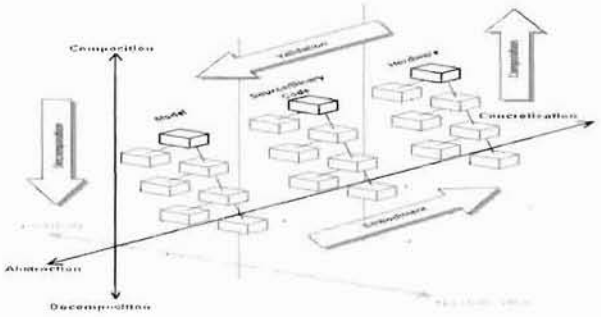

Fig. 1: Development activities in MARMOT

3. PECOS COMPONENT MODEL

PECOS component model was originally developed for field device systems. PECOS component consists of two main parts, i.e. the static structure model which

describes the entities included in the model, their features and properties; and the execution model which defines the behavior of the component execution. The following Subsection describe each parts in detail.

3.1 Static structure model

There are three main entities in the PECOS model:***Components, ports*** and ***connector*** (Genssler et. al, 2002). Components are used to organize the computation and data into parts that have well-defined semantics and behavior. A port is a reference to data that can be read and written by a component and enables a component to be connected to another component through a connector. Data passed over the port is specified with name, type, range and direction (in, out or inout). Only compatible ports can be connected with connectors. Components can be any of three types: active, passive or event component. Active components have their own thread of control; passive components do not have their own thread of control; and an event component is a component that is triggered by event.

PECOS components can be hierarchically built from other subcomponents. A component which contains subcomponents or children is called a composite component or a parent component, and these subcomponents are not visible outside the composite component.

The components composition using PECOS static structural model is performed by connecting the compatible ports between components. Figure 2 shows two active components: MotorSpeedControl and PI marked with "⚒" at the upper right corner, and two passive components: Encoder and Motor.

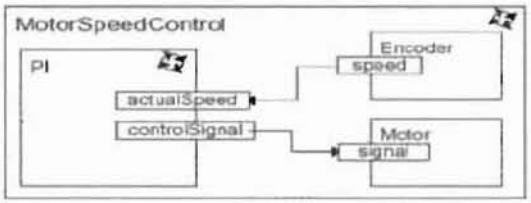

Fig. 2: PECOS structure model for a motor speed control application

3.2 Execution model

There are two different behaviors associated with active and event components: ***execution behavior*** and ***synchronization behavior.*** Execution behavior determines the action that is performed when the component is executed while the synchronization behavior is responsible for synchronizing the data space of the active or event component with that of the parent (Genssler et. al, 2002).

4. PROCESS MODEL PHASE

This process model proposed includes six phase for ERT system development as shown in Figure 3. The first phase is analysis phase where in this phase developer must model the requirements into use case diagram. At the same level, the use case will be described in detail with description table and interaction model shows use case.

The second phase is early design where in this phase developer must define possible class diagram and the operation specification. Class diagram can be component in this stage. After that, activity diagram must be defined to know more details about data input and output.

At the third phase, containment hierarchy will be produced where many class diagrams can be grouped in one package. In this level, each package must define the in port which shows data receive by the package or out port data that will be send respectively from that package.

The fourth phase is the integration phase where integration diagram will be produced based on previous phase. This integration tries to connect more than one package or component to develop new application.

The fifth phase, composition phase will produce composition diagram that includes the runtime behavior.

Last phase, implementation phase is the code template where user can modify the code to adapt it to the environment as needed.

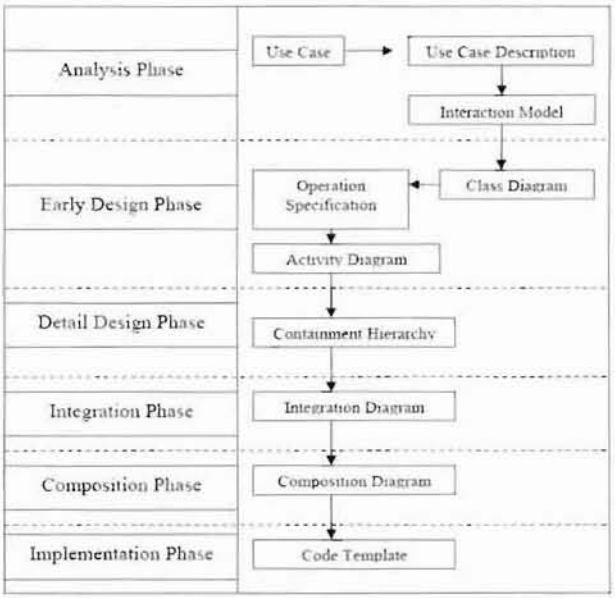

Fig. 3: Process Model Phase for ERT system

5. THE IMR 71848 CASE STUDY

Intelligent Mobile Robot (IMR71848) will be use as a case study in this paper. This robot software project is being prepared as part of a working group in UTM71848RG. The parent project is funded under UTM grant, vot number 71848.

The first phase on the process model proposed earlier, starts with model of the requirement for IMR 71848 using

describes the entities included in the model, their features and properties; and the execution model which defines the behavior of the component execution. The following Subsection describe each parts in detail.

3.1 Static structure model

There are three main entities in the PECOS model:***Components, ports*** and ***connector*** (Genssler et. al, 2002). Components are used to organize the computation and data into parts that have well-defined semantics and behavior. A port is a reference to data that can be read and written by a component and enables a component to be connected to another component through a connector. Data passed over the port is specified with name, type, range and direction (in, out or inout). Only compatible ports can be connected with connectors. Components can be any of three types: active, passive or event component. Active components have their own thread of control; passive components do not have their own thread of control; and an event component is a component that is triggered by event.

PECOS components can be hierarchically built from other subcomponents. A component which contains subcomponents or children is called a composite component or a parent component, and these subcomponents are not visible outside the composite component.

The components composition using PECOS static structural model is performed by connecting the compatible ports between components. Figure 2 shows two active components: MotorSpeedControl and PI marked with "⚡" at the upper right corner, and two passive components: Encoder and Motor.

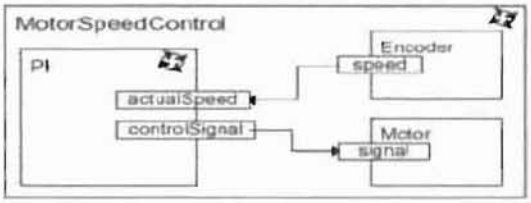

Fig. 2: PECOS structure model for a motor speed control application

3.2 Execution model

There are two different behaviors associated with active and event components: ***execution behavior*** and ***synchronization behavior.*** Execution behavior determines the action that is performed when the component is executed while the synchronization behavior is responsible for synchronizing the data space of the active or event component with that of the parent (Genssler et. al, 2002).

4. PROCESS MODEL PHASE

This process model proposed includes six phase for ERT system development as shown in Figure 3. The first phase is analysis phase where in this phase developer must model the requirements into use case diagram. At the same level, the use case will be described in detail with description table and interaction model shows use case.

The second phase is early design where in this phase developer must define possible class diagram and the operation specification. Class diagram can be component in this stage. After that, activity diagram must be defined to know more details about data input and output.

At the third phase, containment hierarchy will be produced where many class diagrams can be grouped in one package. In this level, each package must define the in port which shows data receive by the package or out port data that will be send respectively from that package.

The fourth phase is the integration phase where integration diagram will be produced based on previous phase. This integration tries to connect more than one package or component to develop new application.

The fifth phase, composition phase will produce composition diagram that includes the runtime behavior.

Last phase, implementation phase is the code template where user can modify the code to adapt it to the environment as needed.

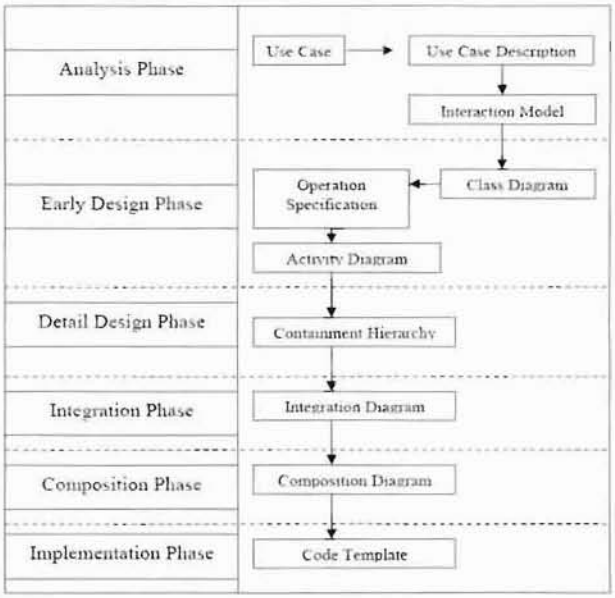

Fig. 3: Process Model Phase for ERT system

5. THE IMR 71848 CASE STUDY

Intelligent Mobile Robot (IMR71848) will be use as a case study in this paper. This robot software project is being prepared as part of a working group in UTM71848RG. The parent project is funded under UTM grant, vot number 71848.

The first phase on the process model proposed earlier, starts with model of the requirement for IMR 71848 using

Use Case Diagram as shown in Figure 4 followed by Use Case Description in Table 1 to elaborate the details of use case. For details about the interaction between user and the system can be seen in Figure 5. Unified Modeling Language (UML) 2.0 is used to design the use case diagram and interaction model.

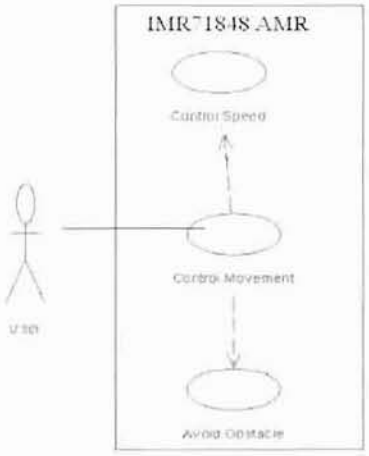

Fig. 4: IMR 71848 Use Case Diagram: Analysis Phase

Table 1 IMR 71848 Use Case Description: Analysis Phase

Name	Avoid Obstacle
Actor	User
Goal	The IMR71848 cruises around the environment to find the passage
Description	The robot needs to avoid obstacles that exist within its environment
Exception	N/A
Rules	N/A
Quality Requirement	N/A
I/O	Input • IR Proximity Output • Motor Speed • Display robot status at LCD

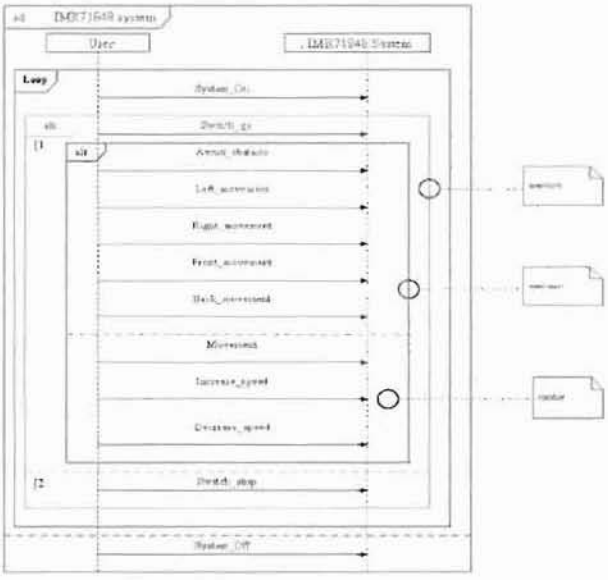

Fig. 5: IMR 71848 Interaction Model: Analysis Phase

At the early design, class diagram as shown in Figure 6 is produced where application IMR 71848 connect with other components such as controller, driver and movement component. In additional, table 2 describe the operation specification that includes description of the operation sensor, constrain, data return and send.

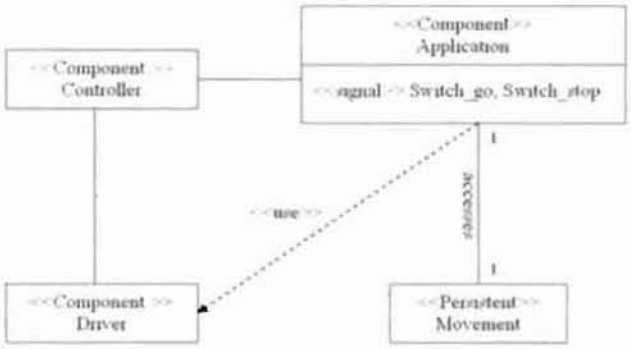

Fig. 6: IMR 71848 Class Diagram: Early Design Phase

Table 2 IMR 71848 Operation Specification: Early Design Phase

Name	Main
Description	Uses two types of sensor: IR Proximity Sensor to detect the presence of obstacles during the movement of the robot and IR Distance Sensor to estimate the distance of obstacles and walls from the robot's current location. Switches: operator switch on, operator switch off, operator switch go and operator switch stop. In addition, encodes change of the tuning pulses that read the speed value of the motor.
Constrain	Encodes 50 millisecond – 100 millisecond Switches 1 sec – 2 sec IR Proximity Sensor 0.5 sec – 1 sec IR Distance Sensor 1 sec – 2 sec
Return	Controls speed and avoids obstacles while displaying information of the status for monitoring purpose
Sends	IR Poximity Sensor IR Distance Sensor Digital camera Motor Encoder Switches LCD RF transceiver

At the detail design phase, knowledge from PECOS component model is used to design the component and MARMOT method is used to identify the relationship between components as shown in Figure 7.

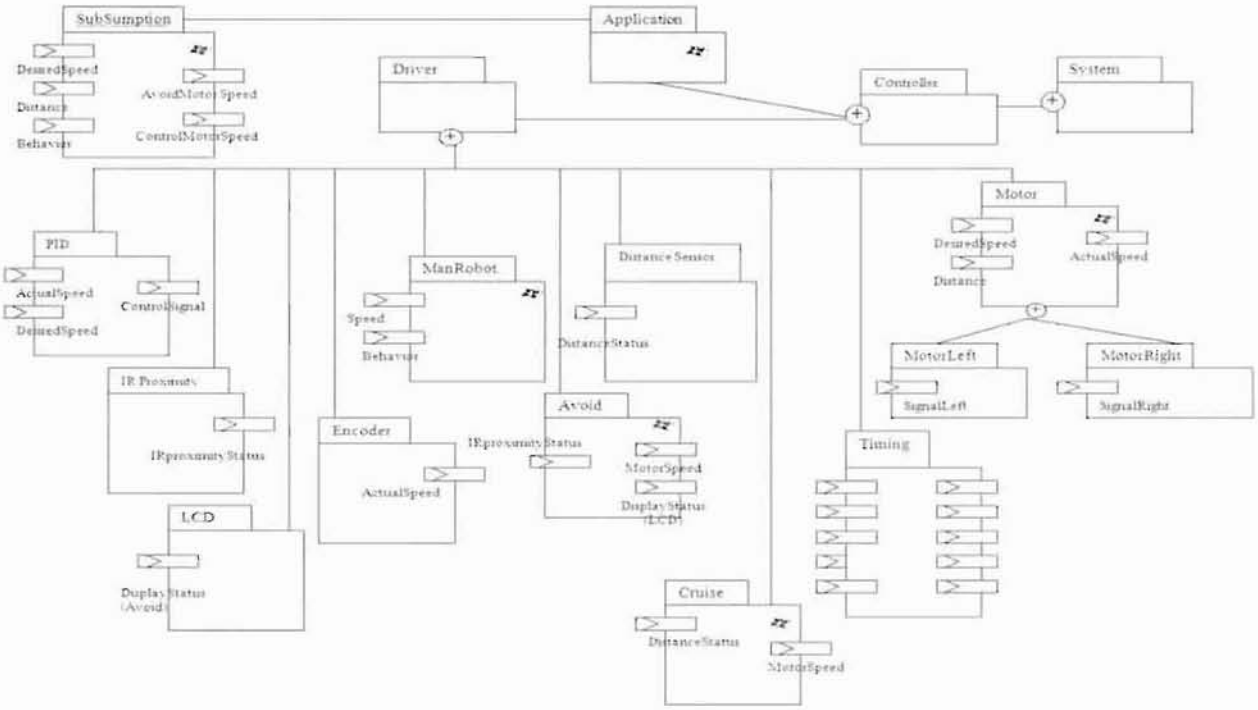

Fig. 7: IMR 71848 Containment Hierarchies

For the integration phase each component that matches will be connected. The connection did not only consider the in / out port only but it is also based on several other rules such as maximum and minimum value, types of data return or send etc.

The MobileRobot component composition of Figure 8 is mapped to a task which executes sequentially two passive components IRSensor and Distancesensor; and six synchronization parts of its child components, i.e. Avoid, Cruise, Subsumption, motorctrl-right, motorctrl-left and manrobotintf,

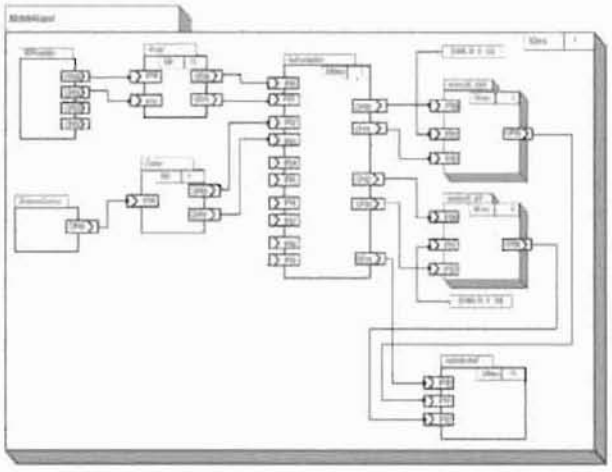

Fig. 8: Components composition of an IMR 71848 application case study

Figure 9 shows the example of template code for the motorControl component.

```
#include "pecos_cfg.h"
#include "pecos.h"
// TOP LEVEL COMPONENT
void MobileRobot (void* data) {
/****** INITIALIZATION PART ******/
  //-- Initialize all child components
  //--   all ACTIVE tasks will be created
    INITmanrobotintf();
    INITmotorctrl_right();
    INITmotorctrl_left();
        :etc
    for (;;)
/****** EXECUTION PART ******/
  //-- Execute all passive components
  //-- Active components already executed
        EXECIRProximity();
        :etc
/****** SYNCHRONIZATION PART ******/
 //-- Synchronous all inports & outports
        SYNCmanrobotintf();
        :etc
//-- Connect all inports to outports
        Avoid.IP00    = IRProximity.OP00;
        Avoid.IP01    = IRProximity.OP01;
        Subsumption.IP00 =    Avoid.OP00;
        Subsumption.IP01[0] = Avoid.OP01[0];
        :etc
//-- call RTK to create periodic execution
        OSTimeDelay (MOBILEROBOT_PERIOD);
}
void main(void) {
    //-- initialize hardware dependent parts
    //-- initialize Real-time kernel
    //-- create MobileRobot task
    //-- start the  Real-time kernel
    while (1) ;          // run endlessly
}
```

Fig. 9: The code template for IMR71848 component task execution

6. DISCUSSION AND CONCLUSION

The main objective of using hybrid component technologies in our work is to support process model flow from analysis to code template for ERT system development. Two component technologies is used in this paper are KobrA and PECOS component technology.

Due to the inability of KobrA to support ERT system, its extension component technology, MARMOT is used in this research. Base on our previous work MARMOT model is good in component modelling and graphical component but it cannot support software component at detail design level and no method interfaces for component communication available. PECOS component technology cannot support component modelling but it good support software component at details design and clearly present interfaces for component communication. PECOS component technology is used to support detail design, integration and composition phase. As known object oriented analysis and design method become matures but for component oriented analysis and design in beginning stage. This paper showed the flow of process model by integrating the two component technologies which are KobrA and PECOS component technologies. Therefore, this process model proposed earlier hopefully can support component based methodological and can support component oriented analysis and design to enable systematic development for ERT system

ACKNOWLEDGEMENTS

Special thanks to Ministry of Science, Technology and Innovation (MOSTI) Science Fund for the financing and funding, Universiti Teknologi Malaysia (UTM) for the facilities and infrastructures, our members software engineering lab 2 for their support and others thanks for Sina Ali for their commitment to revise this paper.

REFERENCES

Atkinson C., Bayer J. Bunse C., Kamsties E., Laitenberger o., Laqua R., Muthig D., Paech B., Wust J. and Zettel J., Component-based Product Line Engineering with UML, *Addison Wisley,* vol. 1 , 2002

Bunse C., Gross H.G. and Peper C., Applying a model based approach for embedded system development, *33rd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2007),* 2007

Bunse. C and Gross. H.G, unifying hardware and software components for embedded system development, *Sprimger-verlag berlin Heidelberg,* pp. 120-136, 2006

Bunse. C, Developing micro-controller-system with UML a MARMOT Case Studty, *IESE-report no.111.06/E,* Version 1.0, 2006

Colin Atkinson, Joachim Bayer, and Dirk Muthing, Component-Based Product Line Development: The KobrA Approach, *Proceedings of the First Software Product Line Conference*, pp. 1-19, 2000.

Genssler. T. et. al., "PECOS in a Nutshell", *Technical Report,* PECOS Project, September 2002.

I-Ling Yen, Jayabharth Goluguri, Farokh Bastanu, Latifur Khan and John Linn, A Component-based Approach for Embedded Software Development, *Proceedings of the Fifth IEEE International Sympoium on Object-Oriented Real-Time Distributed Computing (ISORC '02)*,Pp. 1-8, 2000

Jawawi, D. N. A., Safai Deris and Rosbi Mamat, (2004). "An Evaluation of Embedded Real-Time Component Models for Autonomous Mobile Robot Software Development", National Real-Time Technology And Application Symposium 2004, UPM, KL. Pp. 63-71.

Khan, M.U., Geihs, K., Gutbordt et al, "Model-Driven Development of Real-Time Systems with UML 2.0 and C", *3rd International Workshop an Model-based Methodologies for Pervasive and Embedded Software at the 13th IEEE Int. Conf. on Engineering,* 2006

Martin Torngren, Martin Grimheden and Niklas Adamsson, "Experiences from large embedded systems development projects in education, involving industry and research", *Special issues on the second workshop on embedded system education (WESE),* ACM SIGBED Review, pp. 55-63, 2007

Nierstrasz, O., Arévalo, G., Ducasse, S., Wuyts, R., Black, A., Müller, P., Zeidler, C., Genssler, T., van den Born. R.. A Component Model for Field Devices. *Proceedings First International IFIP/ACM Working Conference on Component Deployment*, Springer-Verlag Heidelberg, Berlin, 200-209, 2002

Ommering, R., Linden, F., Kramer, J. and Magee, J.. The Koala component model for consumer electronics software. IEEE Computer, 33 (3): pp. 78 –85, 2000

Poels G. and Dedene G., Do we need Component-Oriented Analysis & Designmethods?, *Software Practice Advance, AT2000,* 27-29 March 2000.

Stewart, D. B., Volpe, R. A., and Khosla, P. K.. Design of Dynamically Reconfigurable Real-time Software Using Port-Based Objects, *IEEE Transaction on Software Engineering, 23(12):* pp. 759 –776, 1997

Wall A., "Architectural Modeling and Analysis of Complex Real-Time Systems", *Phd Thesis of Malardalen University*, 2003.